Aplikasi *Indonesian News Aggregator* Berbasis Android yang Didukung oleh Sistem Perekomendasi

Nydia Valentina Wahono, Adi Wibowo, Rolly Intan
Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-mail: yuichitaki@yahoo.com, adiw@petra.ac.id, rintan@petra.ac.id

ABSTRAK

Perkembangan teknologi internet berkembang dengan sangat pesat. Hal ini mempengaruhi kebiasaan manusia dalam mencari dan mendapatkan informasi. Berita menjadi tidak dapat dipisahkan dari kehidupan sehari-hari manusia. Ada begitu banyak berita berlimpah dengan berbagai macam topik dan kategori untuk dibaca, sedangkan waktu yang dimiliki oleh manusia untuk membaca sangat terbatas. Oleh karena itu, diperlukan adanya fitur rekomendasi pada aplikasi berita masa kini agar dapat menyuguhkan berita sesuai dengan apa yang disukai dan tidak disukai oleh pengguna, di mana pengguna dapat memberikan *rating* pada setiap berita yang dibaca.

Penelitian ini berfokus pada fitur rekomendasi dari aplikasi berita berbasis Android. Untuk memberikan fitur rekomendasi, diperlukan adanya *server* yang dapat mengambil data berita dari berbagai macam situs. Kemudian, dilakukan *parsing* dan pengolahan pada setiap data berita yang sudah didapatkan untuk dipakai dalam perhitungan profil pengguna.

Berdasarkan penelitian yang sudah dilakukan, pemberian *rating* dapat mengubah rekomendasi berita yang diberikan dari waktu ke waktu dan masih ada beberapa kekurangan pada struktur HTML dan standar RSS dari situs sumber berita berbahasa Indonesia.

Kata Kunci: Algoritma Rocchio, Vector Space Model, Pemrograman *Mobile Device*, Android, Aplikasi Berita, Sistem Perekomendasi.

ABSTRACT

Internet technology has shown its significant growth. It affects human habits in how they seek and gather information. News has become human's daily needs. There are so many kind of news to be read, but time is limited. Therefore, recommendation feature for news feature nowadays is needed, so it is easier to supply what kind of news that readers read, where user can rate the news.

These research focused on recommendation feature for news feature based on Android. To give recommendation feature, there will be a certain server required to grab news data from some specific websites. Then, news data which was grabbed is parsed and processed to be used for members profile calculation.

Based on the research, rating can change news recommendation which is given from time to time and there are still some flaws on HTML structure and RSS standard from Indonesian news websites.

Keywords: *Rocchio Algorithm*, *Vector Space Model*, *Mobile Device Programming*, *Android*, *News Application*, *Recommender System*.

1. PENDAHULUAN

Berita merupakan hal yang tidak dapat dilepaskan dari kehidupan sehari-hari manusia di masa sekarang. Manusia mendapatkan informasi terbaru mengenai banyak hal dari berbagai belahan dunia melalui berita. Perkembangan teknologi ikut berperan penting dalam penggunaan berita di kehidupan sehari-hari. Dulu, berita hanya dimuat pada media cetak, radio, dan televisi. Seiring dengan perkembangan zaman, internet mulai berkembang sehinga akhirnya berita juga dimuat dalam aplikasi *mobile*. Hal ini mengakibatkan terbentuknya tren membaca berita menggunakan peralatan *mobile*.

Beberapa aplikasi berita berbahasa Indonesia yang sudah ada pada peralatan *mobile* berbasis Android adalah Berita Indonesia dan Indonesia News. Aplikasi tersebut menampilkan beberapa sumber berita Indonesia untuk dipilih, setelah itu pengguna dapat membaca semua berita terbaru dari sumber yang sudah dipilihnya. Aplikasi tersebut juga memiliki fitur seperti berita favorit dari yang pernah dibaca dan berita baru yang belum dibaca. Aplikasi di atas menampilkan semua berita yang didapatkan dari situs-situs sumber berita yang sudah ada.

Keterbatasan yang dimiliki oleh kedua aplikasi di atas adalah bahwa berita ditampilkan hanya berdasarkan kategori secara umum saja. Misalnya, seorang pengguna memilih kategori teknologi untuk berita yang ditampilkan. Dalam kategori teknologi, berita mengenai Windows Phone, iOS, Android tentu sama-sama ditampilkan. Padahal pengguna mungkin hanya menginginkan berita mengenai Android saja. Belum ada fitur di mana ada berita-berita yang direkomendasikan khusus sesuai dengan keinginan pengguna dalam membaca berita.

Sebagai solusi untuk permasalahan di atas, dapat digunakan sistem perekomendasi untuk melakukan filter berita berdasarkan keinginan pengguna. Oleh karena itu, penelitian ini mengusulkan aplikasi perekomendasi yang dapat menerima masukan berita-berita yang disukai atau tidak disukai pengguna dan memberikan saran secara otomatis berita-berita apa saja yang sesuai dengan kebutuhan mereka.

2. LANDASAN TEORI

2.1. Vector Space Model

Vector Space Model yang dikemukakan oleh Salton menggabungkan informasi lokal dan global dari kumpulan dokumen. Rumus penghitungan bobot term terhadap sebuah dokumen dapat dilihat secara langsung sebagai berikut [4].

$$Term\ Weight = \quad w_i = tf_i * log\left(\frac{D}{df_i}\right)$$

Keterangan:

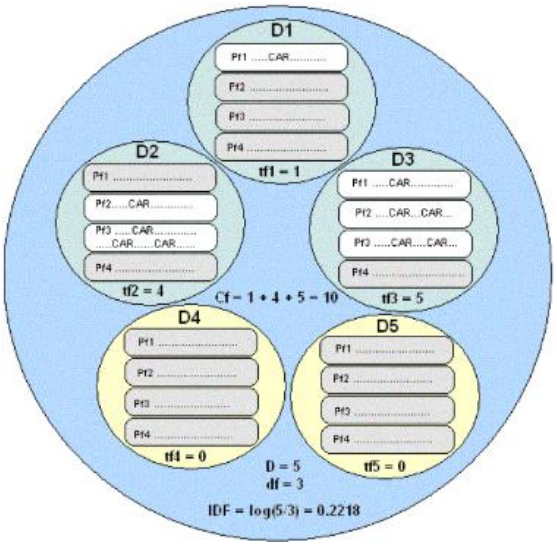
tf_i = *term frequency* (*term counts*) atau berapa banyak sebuah term muncul di sebuah dokumen.

df_i = *document frequency* atau berapa banyak dokumen yang mengandung term i.

D = berapa banyak dokumen dalam sebuah *database*.

Rasio D/df_i merupakan probabilitas dari pemilihan dokumen yang mengandung sebuah query term dari sekumpulan dokumen. Ini dapat dilihat sebagai probabilitas global dari sekumpulan data. Rumus $\log(D/df_i)$ merupakan *inverse document frequency* (IDF_i) dan merupakan informasi global. TF-IDF merupakan pembobotan yang sering digunakan dalam penelusuran informasi [3].

Gambar 1 mengilustrasikan relasi antara frekuensi global dan lokal dalam koleksi *database* ideal yang terdiri dari D_1, D_2, D_3, D_4, dan D_5. Hanya 3 dokumen mengandung term "CAR". Hasil perhitungan *query* dari term ini adalah $\log(5/3) = 0.2218$.



Gambar 1. Ilustrasi Perhitungan *Query* terhadap Dokumen

Sumber: http://www.miislita.com/term-vector/term-vector-3.html

Untuk memahami rumus di atas, digunakan contoh. Untuk menyederhanakan, diasumsikan sedang berurusan dengan *basic term vector model* yang:

a. Tidak memperhitungkan di mana term muncul dalam dokumen.
b. Menggunakan semua term, termasuk term dan *stopwords* yang umum.
c. Tidak mengurangi term pada kata dasarnya (*stemming*).
d. Menggunakan frekuensi mentah untuk term-term dan *queries* (data yang tidak ternomalisasi).

Contoh ini dikemukan oleh Profesor David Grossman dan Ophir Frieder dari Ilinois Institute of Technology untuk memahami bagaimana perhitungan vektor term dilakukan, seperti yang dapat dilihat pada Gambar 2.

Contoh:
D1: "*Shipment of gold damaged in a fire*"
D2: "*Delivery of silver arrived in a silver truck*"
D3: "*Shipment of gold arrived in a truck*"

TERM VECTOR MODEL BASED ON $w_i = tf_i * IDF_i$

Query, Q: "gold silver truck"
D_1: "Shipment of gold damaged in a fire"
D_2: "Delivery of silver arrived in a silver truck"
D_3: "Shipment of gold arrived in a truck"
D = 3; IDF = $\log(D/df_i)$

		Counts, tf_i						Weights, $w_i = tf_i * IDF_i$			
Terms	Q	D_1	D_2	D_3	df_i	D/df_i	IDF_i	Q	D_1	D_2	D_3
a	0	1	1	1	3	3/3 = 1	0	0	0	0	0
arrived	0	0	1	1	2	3/2 = 1.5	0.1761	0	0	0.1761	0.1761
damaged	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
delivery	0	0	1	0	1	3/1 = 3	0.4771	0	0	0.4771	0
fire	0	1	0	0	1	3/1 = 3	0.4771	0	0.4771	0	0
gold	1	1	0	1	2	3/2 = 1.5	0.1761	0.1761	0.1761	0	0.1761
in	0	1	1	1	3	3/3 = 1	0	0	0	0	0
of	0	1	1	1	3	3/3 = 1	0	0	0	0	0
silver	1	0	2	0	1	3/1 = 3	0.4771	0.4771	0	0.9542	0
shipment	0	1	0	1	2	3/2 = 1.5	0.1761	0	0.1761	0	0.1761
truck	1	0	1	1	2	3/2 = 1.5	0.1761	0.1761	0	0.1761	0.1761

Gambar 2. Hasil Perhitungan dari 3 Dokumen

Sumber: http://www.miislita.com/term-vector/term-vector-3.html

Hasil analisa data mentah, kolom demi kolom dari contoh yang dikemukakan oleh Dr. Grossman:

a. Kolom 1-5 → Membuat indeks term-term dari setiap dokumen dan menentukan *term counts* tf_i untuk query dan setiap dokumen D_j.
b. Kolom 6-8 → Menghitung *document frequency* d_i untuk setiap dokumen. Karena $IDF_i = \log(D/df_i)$ dan D=3, dilakukan perhitungan langsung.
c. Kolom 9-12 → Ambil hasil perkalian tf*IDF dan hitung *term weights*. Kolom tersebut dapat dilihat sebagai *sparse matrix* di mana sebagian besar isinya adalah 0.

Dengan tujuan menganalisis kesamaan antar dokumen dan *query*, maka dilakukan penghitungan panjang vektor (abaikan term *zero*) untuk setiap dokumen dan *query* yang ada. $|D_i|$ merupakan panjang vektor dokumen. $|Q|$ merupakan panjang vektor *query*.

$$|D_1| = \sqrt{0.4771^2 + 0.4771^2 + 0.1761^2 + 0.1761^2} = 0.7192$$
$$|D_2| = \sqrt{0.1761^2 + 0.4771^2 + 0.9452^2 + 0.1761^2} = 1.0955$$
$$|D_3| = \sqrt{0.1761^2 + 0.1761^2 + 0.1761^2 + 0.1761^2} = 0.3522$$

$$\therefore |D_i| = \sqrt{\sum_i w_{i,j}^2}$$

$$|Q| = \sqrt{0.1761^2 + 0.4771^2 + 0.1761^2} = 0.5382$$

$$\therefore |Q| = \sqrt{\sum_i w_{Q,j}^2}$$

Selanjutnya, hitung semua *dot producs* (semua *zero products* diabaikan).

$$Q \cdot D_1 = 0.1761 * 0.1761 = 0.0310$$
$$Q \cdot D_2 = 0.4771 * 0.9542 + 0.1761 * 0.1761 = 0.4862$$
$$Q \cdot D_3 = 0.1761 * 0.1761 + 0.1761 * 0.1761 = 0.0620$$

$$\therefore Q \cdot D_i = \sum_i w_{Q,j}\, w_{i,j}$$

Kemudian dapat dihitung *similarity values* berdasarkan nilai yang sudah dihitung sebelumnya [5].

$$\cos\theta_{D_1} = \frac{Q \cdot D_1}{|Q| * |D_1|} = \frac{0.0310}{0.5382 * 0.7192} = 0.0801$$
$$\cos\theta_{D_2} = \frac{Q \cdot D_2}{|Q| * |D_2|} = \frac{0.4862}{0.5382 * 1.0955} = 0.8246$$
$$\cos\theta_{D_3} = \frac{Q \cdot D_3}{|Q| * |D_3|} = \frac{0.0620}{0.5382 * 0.3522} = 0.3271$$
$$\therefore \cos\theta_{D_i} = Sim(Q, D_i)$$
$$\therefore Sim(Q, D_i) = \frac{\sum_i w_{Q,j}\, w_{i,j}}{\sqrt{\sum_j w_{Q,j}^2}\sqrt{\sum_j w_{i,j}^2}}$$

Terakhir, urutkan dan beri peringkat pada setiap dokumen secara *descending* menurut hasil perhitungan *similarity values*.
Peringkat 1: Dokumen 2 = 0.8246.
Peringkat 2: Dokumen 3 = 0.3271.
Peringkat 3: Dokumen 1 = 0.0801.

2.2. Rocchio Algorithm

Algoritma Rocchio berdasarkan pada metode *relevance feedback*. Seperti sistem retrieval informasi lainnya, pendekatan Rocchio *feedback* dikembangkan menggunakan Vector Space Model [1]. Algoritma Rocchio didasarkan pada asumsi bahwa sebagian besar pengguna memiliki konsep umum dimana dokumen harus dinotasikan sebagai relevan atau tidak relevan. Selanjutnya, *query* yang dicari pengguna direvisi untuk memasukkan *arbitrary percentage* dari dokumen relevan dan tidak relevan dengan maksud meningkatkan keakuratan mesin pencari serta presisi. Jumlah dari dokumen relevan dan tidak relevan menyebabkan sebuah *query* yang terbentuk dipengaruhi oleh bobot koefisien a b, c dalam rumus Rocchio. Berikut ini merupakan rumus Rocchio [6].

$$\overrightarrow{Q_m} = \left(a \cdot \overrightarrow{Q_0}\right) + \left(b \cdot \frac{1}{|D_r|} \cdot \sum_{\overrightarrow{D_j} \in D_r} \overrightarrow{D_j}\right) - \left(c \cdot \frac{1}{|D_{nr}|} \cdot \sum_{\overrightarrow{D_k} \in D_{nr}} \overrightarrow{D_k}\right)$$

Keterangan:
$\overrightarrow{Q_m}$ = *modified query vector*
$\overrightarrow{Q_0}$ = *original query vector*
$\overrightarrow{D_j}$ = *related document vector*
$\overrightarrow{D_k}$ = *non-related document vector*
a = *original query weight*
b = *related documents weight*
c = *non-related documents weight*
D_r = *set of related documents*
D_{nr} = *set of non-related documents*

Seperti yang didemonstasikan pada rumus Rocchio, koefisien a, b, c bertanggung jawab untuk membentuk *weight* dari vektor yang dimodifikasi, baik semakin mendekat atau semakin menjauh, dari *original query*, dokumen yang berhubungan, dan dokumen yang tidak berhubungan. Secara khusus, koefisien b dan c dapat dikurangi atau ditambahi secara proporsional pada satu set dokumen yang diklasifikasi oleh pengguna.

2.3. Normalisasi TF-IDF

Perhitungan bobot term terhadap dokumen menggunakan rumus TF-IDF. Metode ini bergantung pada jumlah kemunculan sebuah term dalam dokumen dan *inverse* frekuensi dokumen yang mengandung term tersebut.

Berdasarkan rumus w = TF x IDF, di mana IDF = log(N/n) maka berapapun besar nilai TF, jika N = n, hasilnya akan selalu 0 untuk perhitungan IDF. Untuk mengatasi, maka dapat ditambahkan nilai 1 pada sisi IDF. Selain itu, agar menghasilkan bobot dengan standarisasi angka di antara 0 hingga 1, dilakukan normalisasi terhadap rumus TF-IDF tersebut [2].

Hasil akhir modifikasi rumus TF-IDF yang akan digunakan pada penelitian ini dapat dilihat sebagai berikut:

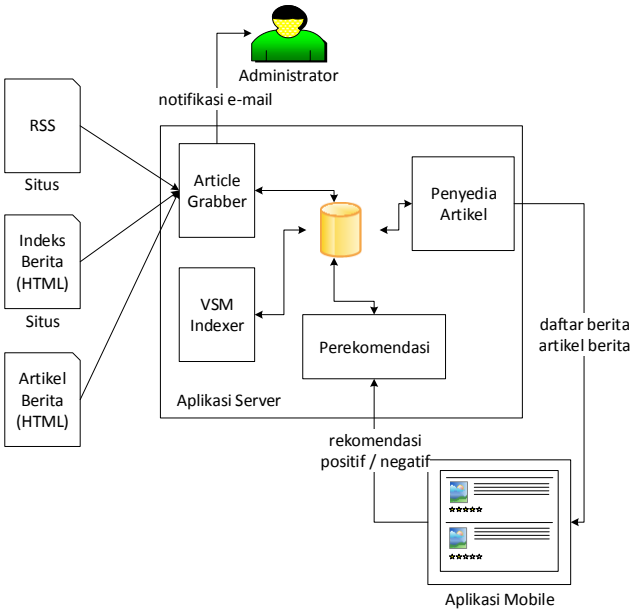
$$w_{ij} = \frac{tf_{ij} \times \left(log\left(\frac{N}{n}\right) + 1\right)}{\sqrt{\sum_{k=1}^{t}(tf_{ik})^2 \times \left[log\left(\frac{N}{n}\right) + 1\right]^2}}$$

Keterangan:
w_{ij} = bobot term t_j terhadap dokumen d_i
tf_{ij} = jumlah kemunculan term t_j dalam d_i
N = jumlah semua dokumen yang ada dalam *database*
n = jumlah dokumen yang mengandung term t_j (minimal ada satu kata yaitu term t_j)

3. DESAIN SISTEM

3.1. Blok Diagram Aplikasi

Blok diagram aplikasi untuk menjelaskan desain sistem dapat dilihat pada Gambar 3.



Gambar 3. Blok Diagram Aplikasi

Aplikasi dibagi menjadi dua bagian, yaitu aplikasi pada sisi *server* dan *mobile*. Aplikasi server dibagi lagi menjadi empat bagian kecil, yaitu *article grabber*, VSM *indexer*, *article provider*, dan *recommender*.

Pada *server*, *article grabber* akan mengambil berita dari beberapa situs, yang dibagi menjadi situs yang mendukung RSS dan tidak mendukung RSS, kemudian memasukkannya ke dalam *database*. Dalam *article grabber* juga terdapat fitur pengiriman *email* notifikasi berisi ID dan URL RSS kepada administrator jika ditemukan artikel kosong pada saat proses *grabbing* dan *parsing*

berlangsung. VSM *indexer* akan melakukan *parsing* pada data berita yang sudah dikumpulkan oleh *article grabber* berdasarkan tipe *file*. Berita tersebut akan diproses sehingga menghasilkan term-term unik agar dapat dilakukan perhitungan VSM dan semuanya disimpan ke dalam *database*. *Article grabber* dan VSM *indexer* dijalankan melalui *console* dan *background process* di server. *Article provider* menyediakan format berita yang akan ditampilkan pada *mobile application* berdasarkan data berita yang sudah disimpan pada *database*. *Recommender* berfungsi untuk memberikan berita yang sesuai dengan profil pengguna, berdasarkan respon suka atau tidak suka yang diberikan pengguna pada saat membaca suatu berita, dan melakukan *update* perhitungan Rocchio terbaru pada *database*.

Mobile application akan meminta pengguna yang pertama kali menggunakan aplikasi untuk melakukan registrasi terlebih dahulu, dengan meminta masukan berupa *username, password*, dll. Setelah proses registrasi dilakukan, pengguna harus melakukan proses *login*. Hal ini untuk membedakan profil setiap pengguna yang tersimpan di *database*. Pada saat pertama kali *login*, pengguna akan diminta memilih beberapa kategori yang disukai, kemudian akan diberikan berita-berita terbaru hari itu, berdasarkan kategori berita yang sudah dipilih. Pada saat pengguna membaca salah satu berita, pengguna dapat memberikan respon pada berita yang sudah dibaca, yaitu memberikan penilaian sesuai dengan masukan yang disediakan. Setelah itu, *mobile application* akan mengirimkan dan menyimpan respon tersebut pada *database* di sisi *server*. Pada kesempatan berikutnya, jika pengguna menggunakan aplikasi, maka aplikasi akan menunjukkan fitur rekomendasi berita berdasarkan respon yang sebelumnya sudah diberikan oleh pengguna.

4. IMPLEMENTASI SISTEM

4.1. *Grabber*

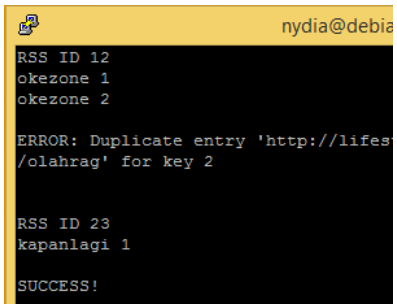
Grabber berguna untuk mengambil data dari berita yang disediakan oleh situs sumber berita. *Grabber* akan mengambil *link* RSS yang sudah tersimpan pada *database* seperti yang terlihat pada Gambar 4.

id_rss	id_situs	id_kategori	url
1	1	1	http://rss.detik.com/index.php/detikcom
2	1	14	http://rss.detik.com/index.php/finance
3	1	5	http://rss.detik.com/index.php/hot
4	1	4	http://rss.detik.com/index.php/inet
5	1	11	http://rss.detik.com/index.php/sport/
6	1	6	http://rss.detik.com/index.php/otomotif

Gambar 4. *Link* RSS yang sudah Disimpan dalam *Database*

Grabber akan mengunduh konten berita yang terdapat pada halaman RSS atau indeks berita yang tersedia. Setelah itu, *grabber* menguduh konten halaman artikel berita berdasarkan *link* yang didapat dari halaman RSS dan indeks berita.

G*rabber* akan berjalan sesuai dengan *scheduler* yang sudah diatur pada *server*. Pada saat *grabber* berjalan, maka pada hasil *output* pada *server* akan terlihat seperti Gambar 5.



Gambar 5. Hasil *Output Grabber* pada *Server*

Berita yang sudah diunduh oleh *grabber* akan disimpan pada tabel berita di *database.* Pada Gambar 6, dapat dilihat berita yang sudah berhasil terambil.

title	link	description
Mantan Menhub Budi Mulyawan: Kalau Regulator Bersi...	http://detik.feedsportal.com /c/33613/f/656082/s/42...	<img src="http://images.detik.com /content/2015/01/...
Kenali 5 Jenis Busi Sepeda Motor	http://detik.feedsportal.com /c/33613/f/656082/s/42...	<img src="http://images.detik.com /content/2015/01/...
The Holy Crab Siap Manjakan Penikmati Seafood di B...	http://lifestyle.okezone.com /read/2015/01/10/298/1...	Menyambut 2015, The Holy Crab akan membuka gerainy...

Gambar 6. Hasil *Grabbing* pada *Database*

4.2. *Indexing*

Indexing terhadap data pada tabel berita berjalan setiap hari sesuai dengan *scheduler*. Hal ini dapat dilihat dari bagaimana sebuah berita yang sudah diambil, melalui *proses parsing*, *check stopword*, *check irregular verb*, *check language*, *remove sign*, dengan proses penghitungan bobot term, dan panjang vektor dokumen, serta pemrosesan terhadap setiap *rating* pengguna terhadap berita sesuai dengan algoritma Rocchio.

Pada Gambar 7, dapat dilihat berita dengan id_536687 berita untuk melalui proses penghitungan.

title	link	description	artikel
Batik Mega Mendung Diklaim Turki, Kemungkinan Stra...	http://lifestyle.okezone.com /read/2015/01/09/194/1...	Turki dikabarkan mengklaim motif batik mega mendun...	<p> Indonesia dihebohkan kabar bahwa Turki m...

Gambar 7. Berita dengan id_berita 536687

Proses yang dilakukan oleh prepare.php adalah mengolah berita berdasarkan Vector Space Model sehingga setiap berita dapat dipotong-potong berdasarkan term yang muncul dari judul, deskripsi, dan artikel berita, kemudian dihitung frekuensi setiap term dalam berita (TF). Dari TF, maka dapat dihitung IDF suatu term. Pada akhirnya, bobot suatu term terhadap dokumen dapat dihitung dari TF dan IDF dan melalui rumus normalisasi seperti yang terlihat pada Gambar 8.

id_berita	id_term	term	tf	idf	weight	cekterm
53668	186590	yofasat	1	4.685007	1.00000	1
53668	186587	pemasaranturki	1	4.685007	1.00000	1
53668	186589	1090070	2	4.383977	0.89443	1
53668	186588	yosafat	4	4.383977	0.89443	1
53668	186056	elyse	2	4.383977	0.89443	1
53668	155251	champ	2	4.082947	0.75593	1
53668	186059	definitli	1	4.383977	0.70711	1
53668	186055	bbrp	1	4.383977	0.70711	1
53668	186061	begituu	1	4.383977	0.70711	1
53668	186060	kyknya	1	4.383977	0.70711	1
53668	186057	masyaallah	1	4.383977	0.70711	1
53668	186058	megemendung	1	4.383977	0.70711	1
53668	68064	latepost	1	4.207886	0.57735	1
53668	16465	kt	1	4.207886	0.57735	1

Gambar 8. Hasil Penghitungan Menggunakan VSM terhadap Berita 53668

Pada saat proses *indexing* terjadi, maka *server* mencetak *output* sesuai dengan proses yang sedang berjalan. Beberapa contoh keluaran *server* dapat dilihat pada Gambar 9 dan Gambar 10.

```
----  27 - kamboja  ----------------------
Lang:
Stem result: kamboja
Inserting kamboja into vsm_term
Success in storing kamboja with TF = 4


----  27 - dilaporkan  -------------------
Lang:
Stem result: dilaporkan
Inserting dilaporkan into vsm_term
Success in storing dilaporkan with TF =


----  27 - terkena  ----------------------
Lang:
Stem result: terkena
Inserting terkena into vsm_term
Success in storing terkena with TF = 4
```

Gambar 9. Hasil Keluaran *Server* untuk Proses *Indexing*

```
nydia@debianx: ~/public_html/ser
_beritaterm
Success in updating weight 0.0149940035976 for berita
_beritaterm
Success in updating weight 0.0149940035976 for berita
_beritaterm
Success in updating weight 0.0149940035976 for berita
_beritaterm
Success in updating weight 0.0149940035976 for berita
_beritaterm
Success in updating weight 0.0149940035976 for berita
_beritaterm
Success in updating weight 0.0149940035976 for berita
_beritaterm
```

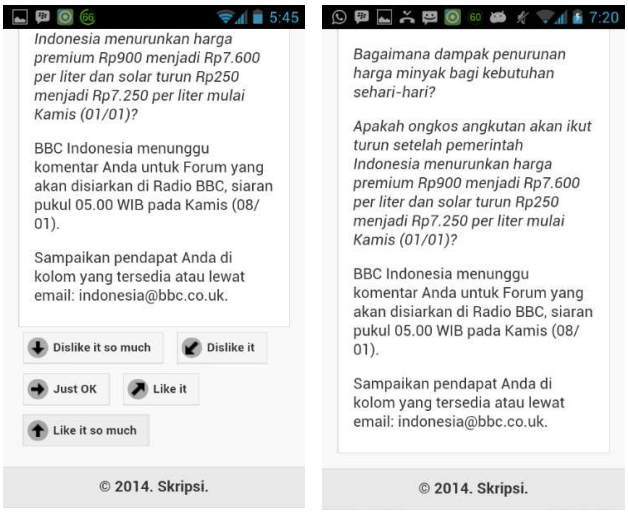
Gambar 10. Hasil Keluaran untuk Proses Penghitungan Bobot Term

4.3. Aplikasi *Mobile*

Pengguna yang baru pertama kali menggunakan aplikasi perlu melakukan registrasi terlebih dahulu. Hal ini untuk membantu pembentukan profil pengguna untuk penghitungan rekomendasi berita berdasarkan algoritma Rocchio.

Pengguna yang sudah melalui proses registrasi dapat melakukan proses *login* dengan memasukkan *username* dan *password* yang sudah terdaftar pada halaman *login*.

Setelah pengguna *login*, maka pengguna dapat memilih berita untuk dibaca kemudian memberikan rating pada berita tersebut. Gambar 11 memperlihatkan bagaimana cara pengguna dapat memberi *rating* terhadap berita, kemudian setelah selesai pemberian *rating*, maka tombol akan hilang.



Gambar 11. Pemberian *Rating* Terhadap Berita

Server akan menyimpan *rating* berita dari pengguna ke dalam tabel member_profile seperti yang terlihat pada Gambar 12.

id_member	id_berita	rate	date	cekrocchio
9	49360	4	2015-01-13 08:45:31	1
9	49740	4	2015-01-13 08:33:37	1
9	49741	4	2015-01-13 08:31:02	1
9	49758	4	2015-01-13 08:33:24	1
9	49768	1	2015-01-13 08:34:03	1
9	49769	1	2015-01-13 08:33:54	1
9	49788	2	2015-01-13 08:34:20	1
9	49928	2	2015-01-13 08:30:55	1
9	49931	5	2015-01-13 08:30:16	1
9	53506	1	2015-01-13 08:31:17	1

Gambar 12. Rating Pengguna id_member 9 terhadap Beberapa Berita

Rating pengguna terhadap berita akan diproses berdasarkan pilihan, kemudian dibagi ke dalam 2 tabel berdasarkan nilai *rating*, yaitu tabel member_related dan tabel member_nonrelated, seperti yang terlihat pada Gambar 13 dan Gambar 14. Hal ini bertujuan untuk

proses penghitungan rekomendasi berita menggunakan algoritma Rocchio.

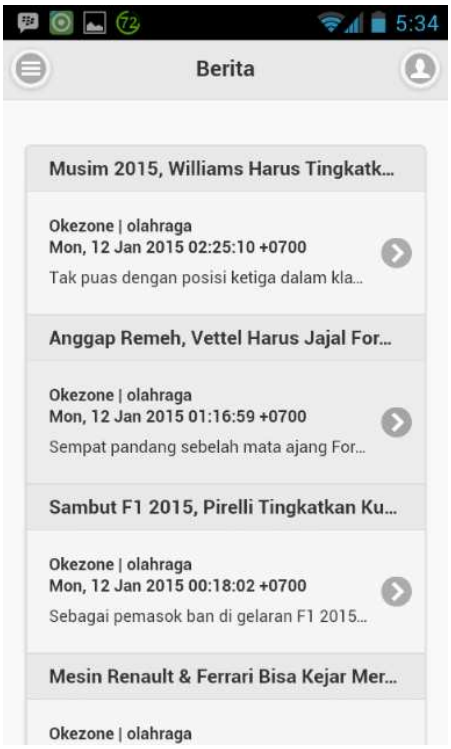
id_member	id_berita	id_term	weight
9	49360	921	0.14072
9	49360	2360	0.11538
9	49360	2545	0.14213
9	49360	2658	0.17263
9	49360	3058	0.10153
9	49360	3064	0.35355
9	49360	3380	0.09029
9	49360	8585	0.20908
9	49360	8596	0.19612

Gambar 13. Tabel member_related Hasil Proses *Rating*

id_member	id_berita	id_term	weight
9	53506	11405	-0.37140
9	53506	11416	-0.46996
9	53506	11417	-0.37140
9	53506	11427	-0.37140
9	53506	11428	-0.37796
9	53506	11430	-0.39056
9	53506	23514	-0.75592
9	53506	54215	-2.00000
9	53506	55748	-0.78446

Gambar 14. Tabel member_nonrelated Hasil Proses *Rating*

Setelah proses penghitungan dokumen relevan dan non relevan dilakukan, maka pada saat pengguna *login* di kesempatan berikutnya, halaman awal aplikasi akan berisi rekomendasi berita sesuai profil pengguna. Halaman awal sebelum proses penghitungan dilakukan dapat dilihat pada Gambar 15. Halaman rekomendasi yang berubah sesuai profil pengguna dapat dilihat pada Gambar 16.



Gambar 15. Halaman Awal Berita sebelum Rekomendasi

Gambar 16. Halaman Awal Berita dengan Rekomendasi

5. KESIMPULAN

Dari proses perancangan, pembuatan, dan hasil pengujian aplikasi *Indonesian News Aggregator* berbasis Android dengan sistem perekomendasi, dapat diambil kesimpulan sebagai berikut:

a. Tidak semua RSS mengacu pada standar RSS terbaru yang valid. Hal ini menyebabkan adanya program *parser* yang dibuat secara spesifik untuk halaman indeks berita tertentu.
b. Tidak semua situs berita menerapkan *semantic* HTML yang benar. Hal ini ditunjukkan dengan tidak adanya identitas judul, isi artikel, penulis berita, dll. pada elemen-elemen HTML. Sebagian situs, seperti www.detik.com, membedakan nama penulis dengan isi berita menggunakan *tag* <b>. Hal ini membuat manusia dapat memahami *tag* tersebut sebagai penulis, namun program tidak dapat membedakan apakah *tag* tersebut merupakan penulis atau kalimat lainnya.
c. Hasil pengujian menunjukkan bahwa pemberian rating, baik dalam nilai positif maupun negatif dapat mengubah berita yang ditampilkan sesuai pemberian rating dari pengguna.

6. DAFTAR PUSTAKA

[1] Fatmawati, T. 2014. *Rocchio Classification*. URI= http://web.unair.ac.id/admin/file/f_41399_RocchioClassification.pdf

[2] Intan, R. & Defeng, A. 2006. HARD:Subject-based Search Engine menggunakan TF-IDF dan Jaccard's Coefficient. *Jurnal Teknik Industri* 8 (1), 61-72. URI=http://fportfolio.petra.ac.id/user_files/92-008/Rolly-TI-Jurnal-June%202006%28new%29.pdf

[3] Isa, T.M., & Abidin, T.F. 2013. Mengukur Tingkat Kesamaan Paragraf Menggunakan Vector Space Model untuk Mendeteksi Plagiarisme. *Seminar Nasional dan Expo Teknik Elektro 2013*, 229-234. URI= http://www.informatika.unsyiah.ac.id/tfa/pdf/papers/SNETE-2013.pdf

[4] Matta, D. & Verma, M. 2013. Evaluating Relevancy of Words in Document Queries Using Vector Space Model. *Journal of Engineering, Computers & Applied Sciences (JEC&AS)* 2, 6 (June 2013), 106-108. URI= http://borjournals.com/Research_papers/Jun_2013/1358IT.pdf

[5] Tudesman, Oktaline, E., Tinaliah, & Yoannita. 2014. Sistem Deteksi Plagiarisme Dokumen Bahasa Indonesia Menggunakan Metode Vector Space Model. *eprints STMIK GI MDP & MDP BUSINESS SCHOOL*. URI= http://eprints.mdp.ac.id/998/1/21tudesmanJurnal.pdf

[6] Yugianus, P., Dachlan, H.S., & Hasanah, R.N. 2013. Pengembangan Sistem Penelusuran Katalog Perpustakaan Dengan Metode Rocchio Relevance Feedback. *Jurnal EECCIS* 7, 1 (Juni 2013), 47-52. URI=http://jurnaleeccis.ub.ac.id/index.php/eeccis/article/viewFile/201/174